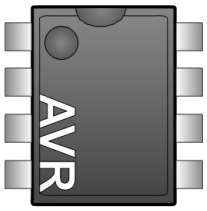


AVRPascal

Manual

version 12/14/2024

1. Introduction

AVRPascal is a Pascal code editor designed for programming AVR microcontrollers (ATtiny and ATmega families). It leverages the Free Pascal[1] compiler to compile source code and AVRdude[2] to transfer the compiled code to the microcontroller's memory using programmers like USBasp[3] or Arduino[4]. Key features:

- Syntax highlighting for improved code readability.
- Error indication: Points to the line of code containing compilation errors.
- Tooltips: Provide information about function and procedure parameters, record fields, and class methods.
- Tabbed document interface for efficient code organization.
- Microcontroller fuse-bit configuration capability.
- Automatic detection of supported USB devices.

AVRPascal is a 64-bit application. It is available in three versions for the following operating systems: Windows, Linux (Debian-based distributions), and MacOS. The functionality of these versions is almost identical, with minor differences arising from the specific requirements of each operating system.

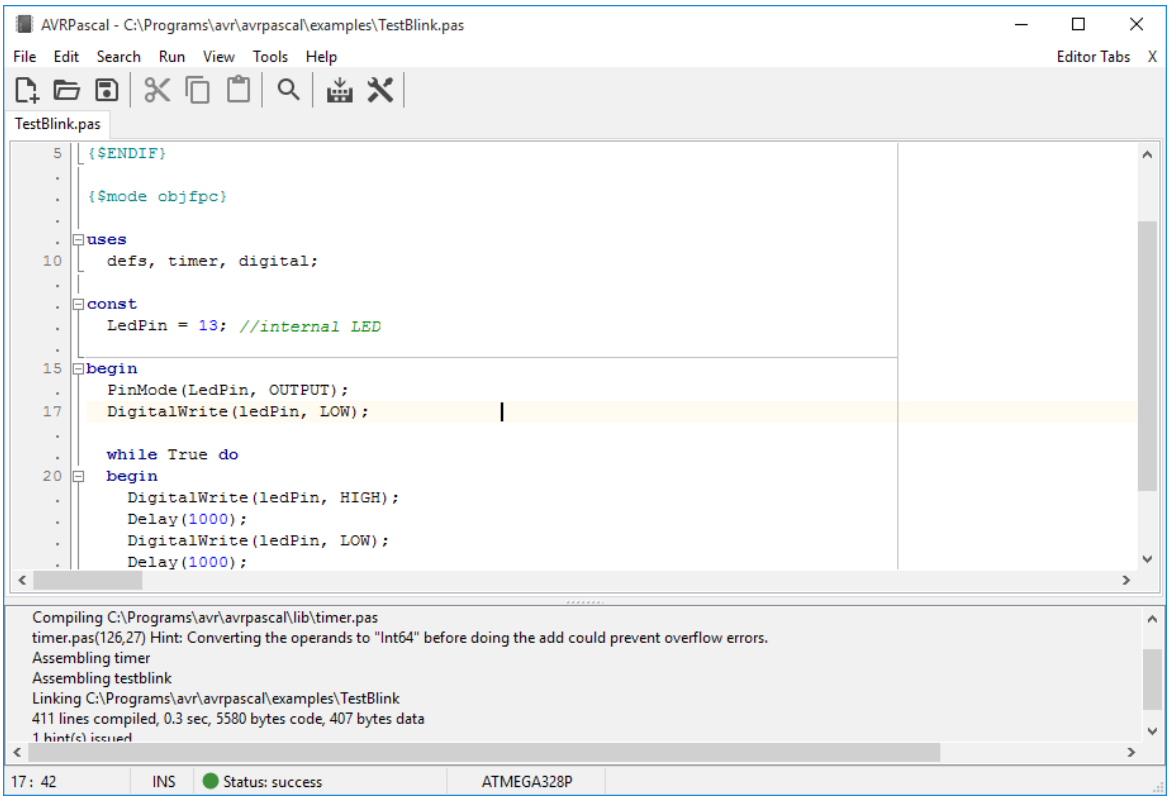
2. Installation

The application should be installed in the target operating system using using the appropriate installer. On Windows use the *.exe* file. You may need to install additional drivers for the USBasp programmer (*ibusb*[5] library driver) or Arduino[6] board. On Linux (Debian-based distributions) use the *.deb* package. Additional drivers are usually not required as they're included in the operating system. For USBasp or Arduino, ensure AVRPascal has administrator privileges or add your user to the *dialout* group. On MacOS use the *.pkg* file. No additional procedures should be necessary.

1 https://www.freepascal.org/
2 https://savannah.nongnu.org/projects/avrdude/
3 https://www.fischl.de/usbasp/
4 https://store.arduino.cc/products/. For a list of supported Arduino boards, see chapter 5.
5 http://zadig.akeo.ie/
6 https://docs.arduino.cc/tutorials/generic/DriverInstallation/

3. Editor

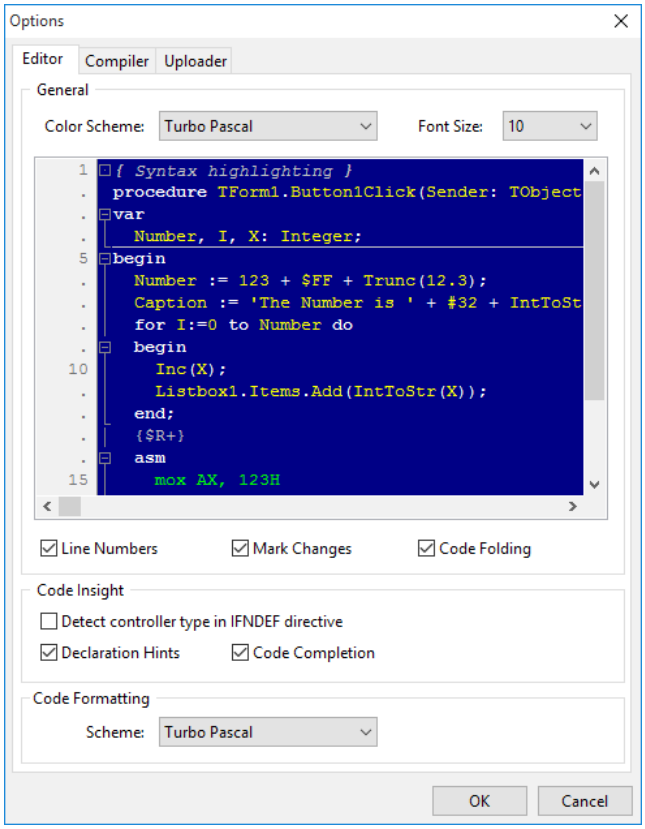
The application window, similar to other code editing and compiling applications, is divided into several areas: menu bar, a toolbar, a tabs area, a messages area, and a status bar. The application allows you to edit multiple files in tabs, with at least one editor tab always open. The message list includes notifications from the compiler, uploader and debugger (if installed).



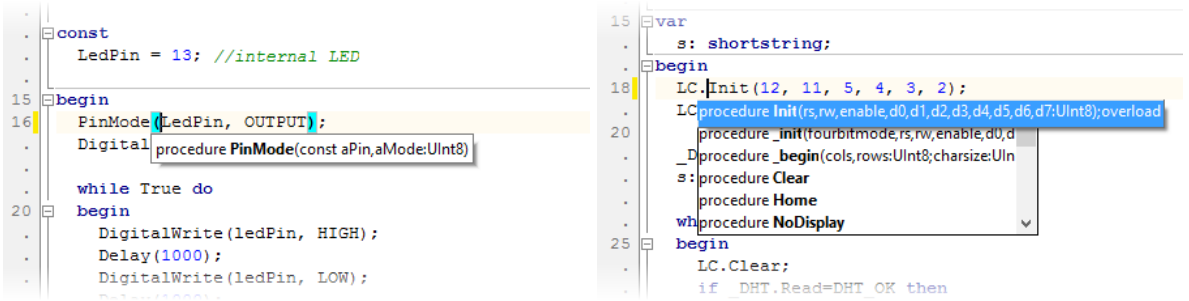
The application's menu offers standard editor functions for opening, editing, and saving Pascal source files (.pas, .pp, .inc). You can also search within the edited file. Additionally, standard text editing functions like selection, cutting, copying, pasting, deleting, and undo/redo are available.

The main feature of AVRPascal that makes working with source code easier is Pascal syntax highlighting. Currently, the application supports three schemes: Delphi, Turbo Pascal (referring to the colors used in these applications) and Twilight (high-contrast colors).. The default color scheme is the *Delphi* scheme. It can be changed by selecting the *Options* menu function and going to the *Editor* tab. Here you can also define font size, and the features of the editor's sidebar (gutter): visibility of line numbers, color highlighting of lines that have been changed and the ability to code folding. Additionally, you can export source code with syntax highlighting to an HTML file (*File->Export to HTML*)[7].

7 The MacOS version does not include the ability to export Pascal source files to HTML format.



AVRPascal further simplifies your work with source code through two features: declaration hints and code completion (list of record type fields and class type methods). These features, common in code editors, provide suggestions as you type. Parameter hints appear after opening a parenthesis (bracket), while a list of record fields and class methods pops up after a dot.



A similar feature is the ability to detect the microcontroller type based on the $IFNDEF or $IF NOT DEFINED directives (see Chapter 4). If this option is enabled, when opening and saving a file and finding such a directive in the source code, a dialog window will appear asking whether to change the current microcontroller type to the one required by this directive.

AVRPascal also has a built-in source code formatter supporting two formatting schemes (styles): *Delphi* and *Turbo Pascal*, defined in the *Options* window, *Editor* tab. The formatter can be invoked by the menu *Edit->Format Source* and formats the text of the active program tab[8].

For increased efficiency, AVRPascal provides a set of keyboard shortcuts for working with source code:

Shortcut	**Function**
Ctrl+N	Opens a new editor tab (*File->New*)
Ctrl+O	Opens the source file (*File->Open*)
Ctrl+S	Saves the source file (*File->Save*)
Ctrl+Z	Undoes the last change in the code (*Edit->Undo*)
Ctrl+X	Cuts the selected text and places it in the clipboard (*Edit->Cut*)
Ctrl+C	Copies selected text (*Edit->Copy*)
Ctrl+V	Pastes the selected text (*Edit->Paste*)
Ctrl+A	Selects the entire text of the document (*Edit->Select All*)
Del	Deletes the selected text or character after the cursor
Ins	Switches between insert and overwrite characters
Ctrl+F	Displays a search dialog for the entered characters in the text (*Search->Find*)
F3	Finds the next occurrence of the entered characters in the text (*Search->Find Next*)
Ctrl+/	Comments on the selected text using “//” characters or removes the comment
Ctrl+U, Tab	Increases the indentation of selected text by two spaces
Ctrl+I, Shift+Tab	Decreases the indentation of selected text by two spaces
Ctrl+Up arrow	Moves the cursor to the procedure/function/method declaration
Ctrl+Down arrow	Moves the cursor to the first line of procedure/function/method code
Ctrl+[text under the mouse cursor]	When clicked, it searches the current document and other documents defined in the uses section for text declarations “under" the mouse cursor. Equivalent to the *Find declaration* context menu function.
Ctrl+ [+ on numeric keypad]	Increases the text font size
Ctrl + [- on numeric keypad]	Decreases the text font size
Ctrl+D	Formats the code according to the settings in the *Options* window,

8 The formatter functionality is based on a modified code of the DelForEx library written by Egbert van Nes (http://www.aew.wur.nl/UK/Delforexp).

	Editor tab, *Code Formatting* frame.
Ctrl+H	Moves the cursor to the previous point in the jump history.
Ctrl+Shift+H	Moves the cursor to the next point in the jump history.

4. Compiling your programs

AVRPascal uses the Free Pascal compiler version 3.3.1. To compile the code, Free Pascal translates Pascal code into assembly code and then utilizes the GNU toolchains for AVR microcontrollers, which includes the avr-as assembler and the avr-ld linker. This process generates three output files: *.bin (binary), *.elf (linker format), and *.hex (text format). Due to Free Pascal's capabilities, the list of supported AVR microcontrollers is quite wide, encompassing the ATtiny and ATmega families:

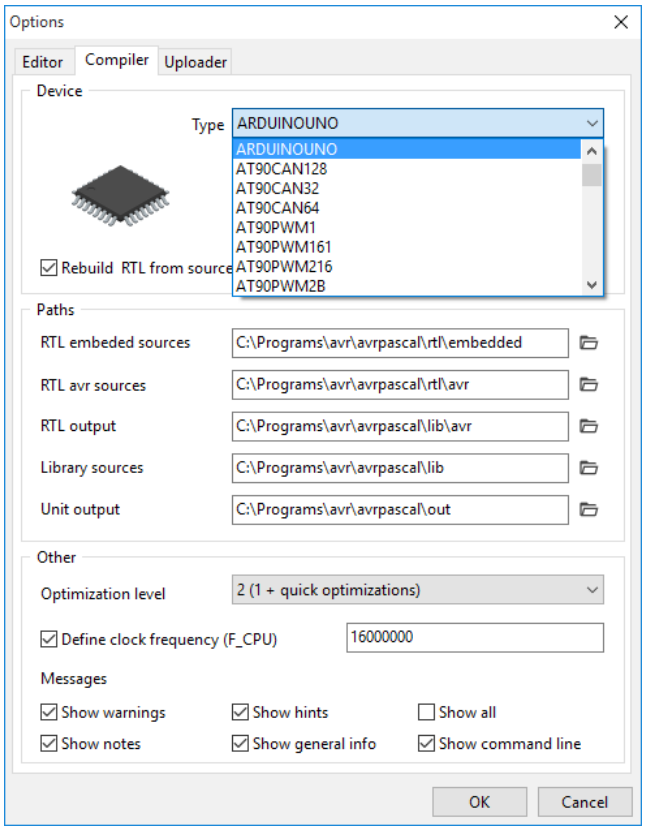
AT90CAN128	*ATMEGA2560*	*ATMEGA645*	*ATTINY212*
AT90CAN32	*ATMEGA2561*	*ATMEGA6450*	*ATTINY214*
AT90CAN64	*ATMEGA2564RFR2*	*ATMEGA6450A*	*ATTINY2313*
AT90PWM1	*ATMEGA256RFR2*	*ATMEGA6450P*	*ATTINY2313A*
AT90PWM161	*ATMEGA32*	*ATMEGA645A*	*ATTINY24*
AT90PWM216	*ATMEGA3208*	*ATMEGA645P*	*ATTINY24A*
AT90PWM2B	*ATMEGA3209*	*ATMEGA649*	*ATTINY25*
AT90PWM316	*ATMEGA324A*	*ATMEGA6490*	*ATTINY26*
AT90PWM3B	*ATMEGA324P*	*ATMEGA6490A*	*ATTINY261*
AT90PWM81	*ATMEGA324PA*	*ATMEGA6490P*	*ATTINY261A*
AT90USB1286	*ATMEGA324PB*	*ATMEGA649A*	*ATTINY28*
AT90USB1287	*ATMEGA325*	*ATMEGA649P*	*ATTINY3214*
AT90USB162	*ATMEGA3250*	*ATMEGA64A*	*ATTINY3216*
AT90USB646	*ATMEGA3250A*	*ATMEGA64C1*	*ATTINY3217*
AT90USB647	*ATMEGA3250P*	*ATMEGA64HVE2*	*ATTINY4*
AT90USB82	*ATMEGA3250PA*	*ATMEGA64M1*	*ATTINY40*
ATA6285	*ATMEGA325A*	*ATMEGA64RFR2*	*ATTINY402*
ATA6286	*ATMEGA325P*	*ATMEGA8*	*ATTINY404*
ATMEGA128	*ATMEGA325PA*	*ATMEGA808*	*ATTINY406*
ATMEGA1280	*ATMEGA328*	*ATMEGA809*	*ATTINY412*
ATMEGA1281	*ATMEGA328P*	*ATMEGA8515*	*ATTINY414*
ATMEGA1284	*ATMEGA328PB*	*ATMEGA8535*	*ATTINY416*
ATMEGA1284P	*ATMEGA329*	*ATMEGA88*	*ATTINY416AUTO*

ATMEGA1284RFR2	*ATMEGA3290*	*ATMEGA88A*	*ATTINY417*
ATMEGA128A	*ATMEGA3290A*	*ATMEGA88P*	*ATTINY4313*
ATMEGA128RFA1	*ATMEGA3290P*	*ATMEGA88PA*	*ATTINY43U*
ATMEGA128RFR2	*ATMEGA3290PA*	*ATMEGA88PB*	*ATTINY44*
ATMEGA16	*ATMEGA329A*	*ATMEGA8A*	*ATTINY441*
ATMEGA1608	*ATMEGA329P*	*ATMEGA8HVA*	*ATTINY44A*
ATMEGA1609	*ATMEGA329PA*	*ATMEGA8U2*	*ATTINY45*
ATMEGA162	*ATMEGA32A*	*ATTINY10*	*ATTINY461*
ATMEGA164A	*ATMEGA32C1*	*ATTINY102*	*ATTINY461A*
ATMEGA164P	*ATMEGA32HVB*	*ATTINY104*	*ATTINY48*
ATMEGA164PA	*ATMEGA32HVBREVB*	*ATTINY11*	*ATTINY5*
ATMEGA165A	*ATMEGA32M1*	*ATTINY12*	*ATTINY804*
ATMEGA165P	*ATMEGA32U2*	*ATTINY13*	*ATTINY806*
ATMEGA165PA	*ATMEGA32U4*	*ATTINY13A*	*ATTINY807*
ATMEGA168	*ATMEGA406*	*ATTINY15*	*ATTINY814*
ATMEGA168A	*ATMEGA48*	*ATTINY1604*	*ATTINY816*
ATMEGA168P	*ATMEGA4808*	*ATTINY1606*	*ATTINY817*
ATMEGA168PA	*ATMEGA4809*	*ATTINY1607*	*ATTINY828*
ATMEGA168PB	*ATMEGA48A*	*ATTINY1614*	*ATTINY84*
ATMEGA169A	*ATMEGA48P*	*ATTINY1616*	*ATTINY841*
ATMEGA169P	*ATMEGA48PA*	*ATTINY1617*	*ATTINY84A*
ATMEGA169PA	*ATMEGA48PB*	*ATTINY1624*	*ATTINY85*
ATMEGA16A	*ATMEGA64*	*ATTINY1626*	*ATTINY861*
ATMEGA16HVA	*ATMEGA640*	*ATTINY1627*	*ATTINY861A*
ATMEGA16HVB	*ATMEGA644*	*ATTINY1634*	*ATTINY87*
ATMEGA16HVBREVB	*ATMEGA644A*	*ATTINY167*	*ATTINY88*
ATMEGA16M1	*ATMEGA644P*	*ATTINY20*	*ATTINY9*
ATMEGA16U2	*ATMEGA644PA*	*ATTINY202*	
ATMEGA16U4	*ATMEGA644RFR2*	*ATTINY204*	

Compiler settings can be found in the *Compiler* tab of the *Options* window (accessible through the *View->Options* menu). These settings are global, meaning any source file compiled in AVRPascal will use the configurations defined here.

In the *Device* frame, choose the target microcontroller type or Arduino board. This ensures specific definitions are included during compilation for the selected microcontroller[9]. The *Rebuild RTL from source if needed* option determines whether AVRPascal automatically rebuilds Free Pascal runtime library (RTL) files after changing the microcontroller type. If unchecked, AVRPascal prompts you for confirmation. Additionally, the *Run->Rebuild RTL Sources* menu allows you to rebuild RTL on demand[10].

The *Paths* frame manages the locations for files used by Free Pascal during compilation. Paths defined in *RTL avr sources*[11] and *RTL embeded sources*[12] determine the source files for the Free Pascal runtime library (RTL), which provide essential functionalities used by compiled programs. *RTL Output* and *Library Sources* paths specifie the locations where the compiled versions of the RTL files or other library files[13] are saved. *Unit Output* path determines where the compiled code for your program files (units) will be placed. The paths are automatically set during AVRPascal installation and changing them is not recommended for beginners as it can affect compilation processes.



9 The current microcontroller type is also displayed on the status bar.

10 In practice, AVRPascal rebuilds three files: system.pp, objpas.pp and the appropriate microcontroller configuration file, e.g. atmega328p.pp. It uses the -O2 optimization level and the -dRELEASE mode.

11 The *RTL avr sources* path is used to store definition files for individual types of AVR microcontrollers.

12 The *RTL embedded sources* path is used to store Free Pascal "system" files from which the system.o file is builded.

13 The *Library sources* path is used to point to the sources of AVRPascal's UnoLib library.

The *Other* frame includes additional compiler settings like optimization level[14], target microcontroller's clock frequency, and the verbosity level of compiler messages displayed in the AVRPascal messages area.

As mentioned earlier, AVRPascal utilizes global compiler settings and doesn't rely on project files or additional definition files. Consequently, it uses the traditional Pascal source file types: defined by *program* keyword, for creating executable files, and defined by *unit* keyword, for creating module files. Moreover during compilation AVRPascal defines a symbol corresponding to the global microcontroller type (e.g. *atmega328p*, see the list above) and a symbol indicating the family it belongs to (*familytiny* or *familymega*)[15]. These symbols enable features like conditional compilation (include or exclude code segments based on the chosen microcontroller) or defining compilation errors (generate specific error messages based on the microcontroller type), e.g.:

```
{$IFNDEF atmega328p}
 {$Fatal Invalid controller type, expected: atmega328p}
{$ENDIF}
```

In this example, if the global microcontroller type is other than *atmega328p*, compilation of the source file will be interrupted and an error message *Invalid controller type, expected: atmega328p* will be displayed in the messages area.

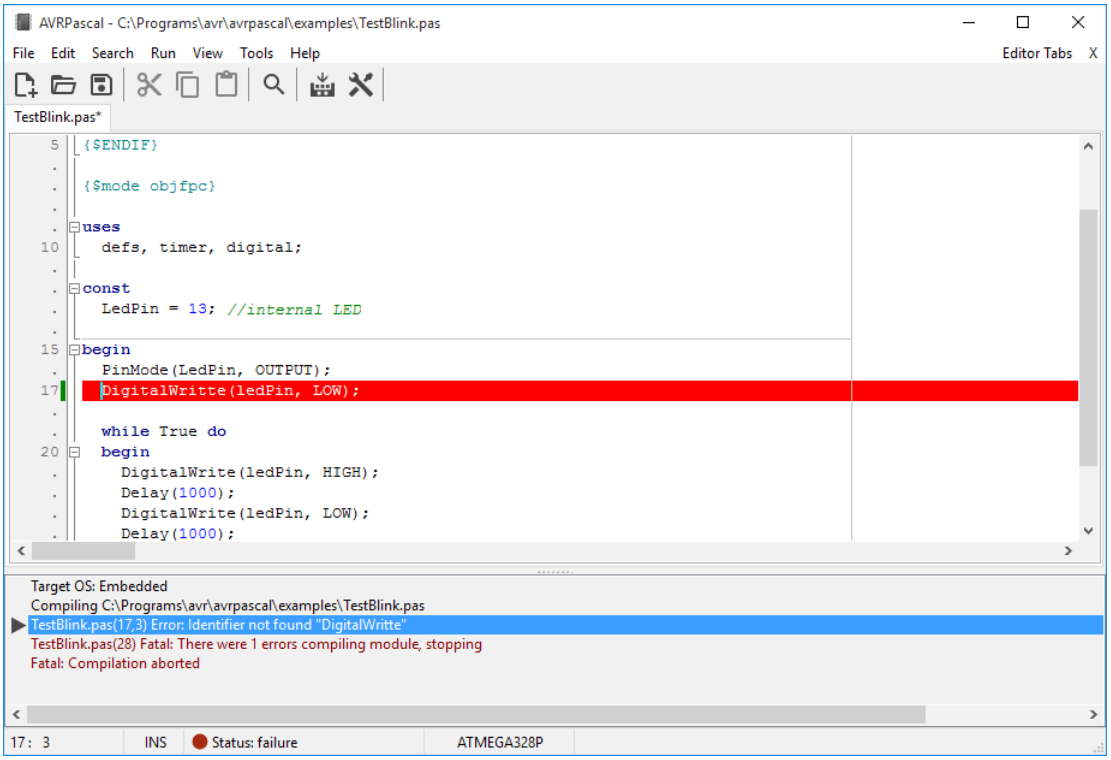
Compilation and execution commands for the currently active editor tab are located in the *Run* menu. They allow compilation (*Compile*), build (*Build All*), deletion of compilation result files (*Clear All*) and rebuilding RTL files depending on selected microcontroller (*Rebuild RTL Sources*). The *Upload* function allows you to transfer the compiled program to the target microcontroller's flash memory using the uploader. This enables running the program directly on the device. AVRPascal also offers keyboard shortcuts to streamline compilation and execution processes:

Shortcut	**Function**
Ctrl+F9	Compiles the source file from the active tab (*Run->Compile*)
Shift+F9	Builds the source file from the active tab (*Run->Build All*)
F9	Exports the compiled source file to the microcontroller's flash memory via the uploader (*Run->Upload*)

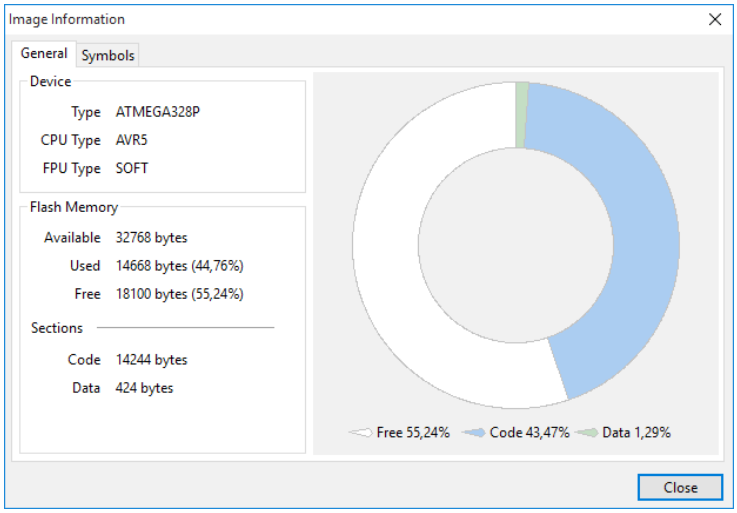
If the compilation was successful, the status bar will display *Status: success*, otherwise *Status: failure*. Additionally, if the compiler indicated an incorrect place in the source code, the line with the error will be highlighted in red.

14 Using high build optimization levels is not recommended, as it may increase the chance of encountering compilation errors or issues with correctly compiling the source files. Moreover, when compiling source files, support for labels (-Sg) and procedure/function inlining (-Si) are enabled by default.

15 If the device type selected by the user is an Arduino board, then the corresponding microcontroller type is additionally defined. Regardless of this, FPC version 3.3.1 defines analogous symbols, preceded by the prefix *FPC_MCU*, e.g. *FPC_MCU_ATMEGA328P*, although in the case of Arduino boards it does not define the corresponding microcontroller types.



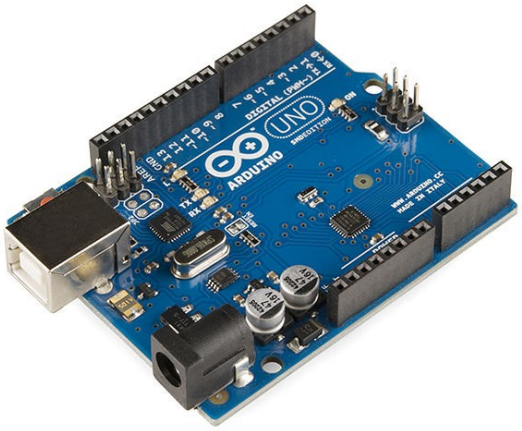
After successful compilation, it is possible to view flash memory usage statistics divided into sections along with a list of symbols generated by the compiler. This data is available in the *Image Information* window (*Run->Image Information…*)[16].



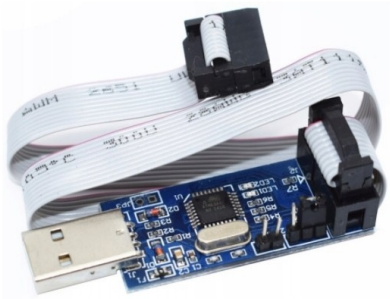
5. Using the programmer and uploader

16 These are code and data sections. However, the total size of both sections reported by the compiler in Messages is slightly larger than the actual size of the code and data in the binary file (*.bin). The difference is that the binary file does not contain the .bss section, which the compiler includes (along with .data) in the data section.

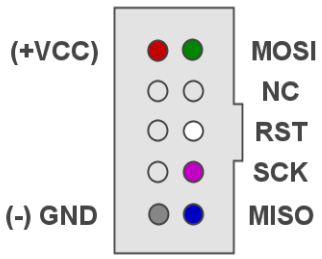
To run the compiled source code, it must be loaded into the microcontroller's flash memory using a programmer, which is a physical device that communicates with the microcontroller. The programmer is typically connected to the computer via a USB port[17]. An uploader, which is a program that sends binary data to the programmer, is used to transfer the compiled code. AVRPascal uses AVRDude in version 8.0 as an uploader, which supports most AVR microcontrollers and works with both the USBasp programmer and Arduino boards.



Arduino Uno



USBasp programmer with tape



USBasp programmer connector

In the case of Arduino used as a programmer, simply connect the device to the computer via the USB port. However, it is important to remember to use sources designed for the microcontroller on which the board is based when programming Arduino. AVR Pascal supports the following Arduino boards (the type of microcontroller of the board is given in brackets):

ARDUINO LEONARDO (ATMEGA32U4)

ARDUINO MEGA (ATMEGA2560)

ARDUINO MICRO (ATMEGA32U4)

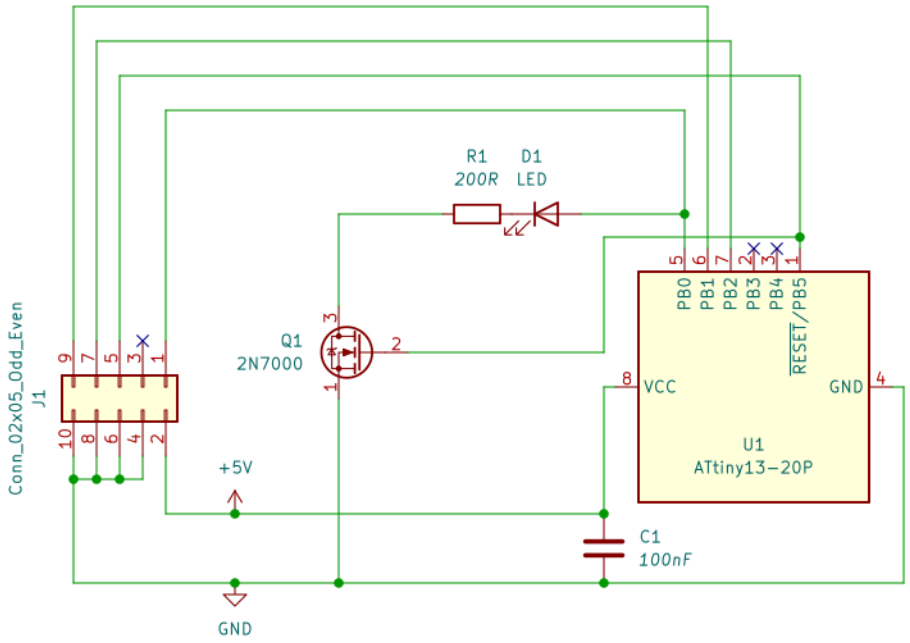
ARDUINO NANO (ATMEGA328P)

ARDUINO NANO EVERY (ATMEGA4809)

ARDUINOUNO (ATMEGA328P)

17 There are other methods used in older generations of PCs, using e.g. the COM or LPT port, but these solutions are impractical because the standard is currently the USB port and few computers have a COM or LPT connection.

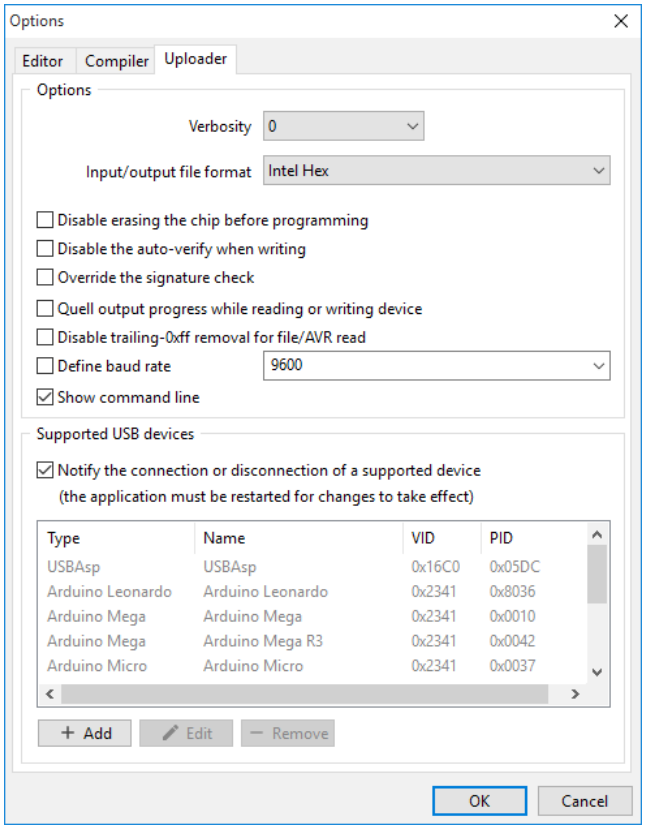
In the case of the USBasp programmer, you must ensure the correct connection of the signal lines (MISO, MOSI, RST, SCK) and power supply (VCC, GND) to the microcontroller. The method of connecting the pins of specific microcontrollers should be checked in their datasheets. For ATtiny13 it may be similar to the following:



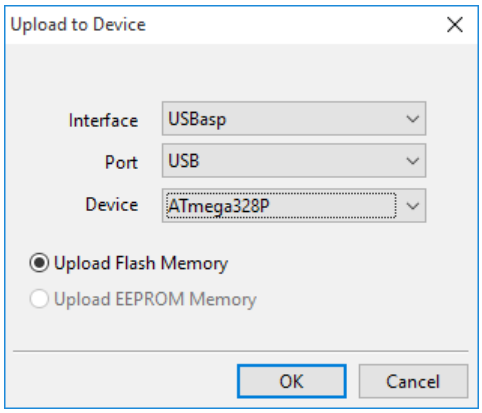
Here, the programmer should be connected to a 10-pin IDC10 (Kanda) connector, pins 4, 6, 8 and 10 of which are connected to ground, pin 2 to the 5V power supply, pin 1 to the PB0 port, pin 5 to PB5, pin 7 to PB2, pin 9 from PB1 of the microcontroller, while pin 3 remained unconnected. The circuit uses a 2N7000 MOSFET transistor that opens if the gate voltage exceeds approximately 2 V. During programming, USBasp shorts pin 1 (PB5) of the microcontroller to ground, closing the transistor and cutting off the line connected to the drain, in this case the LED. The transistor therefore plays the role of a switch that cuts off the ground line of the diode (or other elements connected in this way) during programming, and it does not negatively affect the operation of the programmer. On the other hand, pin 1 (PB5) can be used during normal operation of the microcontroller.

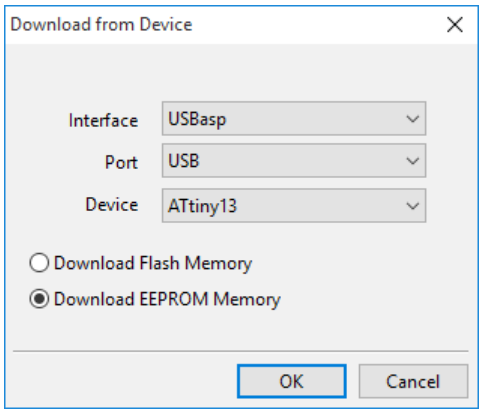
The uploader parameters are located in the *Uploader* tab of the *Options* window (*View->Options* menu). There you can specify the level of detail of the uploader messages displayed in the messages area (level 0 to 4), the format of the input and output files (*.bin, *.elf or *.hex) and the details of AVRdude's operation when programming or reading data from the microcontroller. These include: disabling data clearing mode on the microcontroller before programming, disabling write verification, skipping microcontroller signature checking, visibility of read/write progress from/to the microcontroller, removing trailing 0xFF characters when reading from a file/microcontroller, and setting baud rate[18].

18 In most cases, when working in AVRPascal, changing the uploader parameters is not necessary.



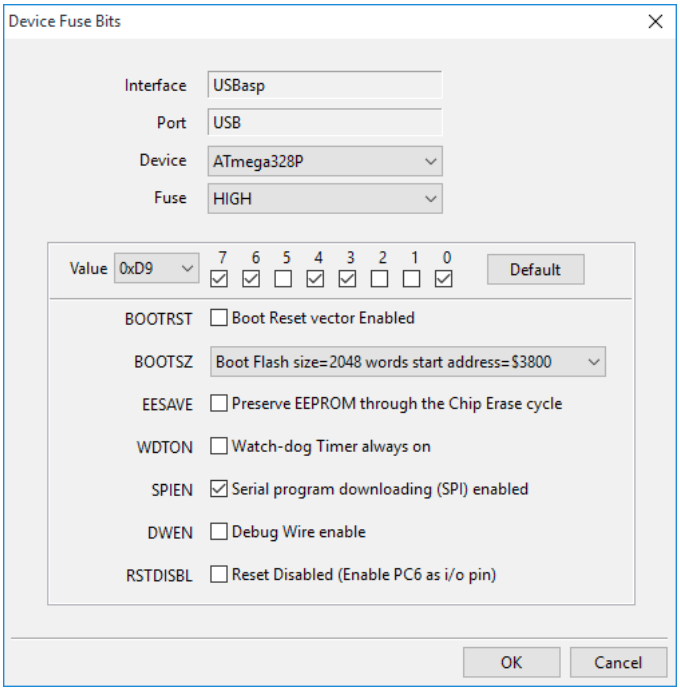
The uploader used in AVRPascal is primarily responsible for saving the compiled program's code (the effect of source compilation) to the microcontroller's flash memory. This can be done using the *Run->Upload* command (see Chapter 4). After running this command, a dialog box appears. The Interface field may show USBasp (if the microcontroller supports the ISP protocol) or Arduino (if one of the listed Arduino boards is set as the target device type for the compiler). In other cases, writing to the microcontroller's flash memory will not be possible. The *Device* field displays a list of subtypes of the selected microcontroller that are supported by the uploader. Similar to working with the compiler, uploader messages are displayed in the messages area and the result of the upload process appears in the status bar.



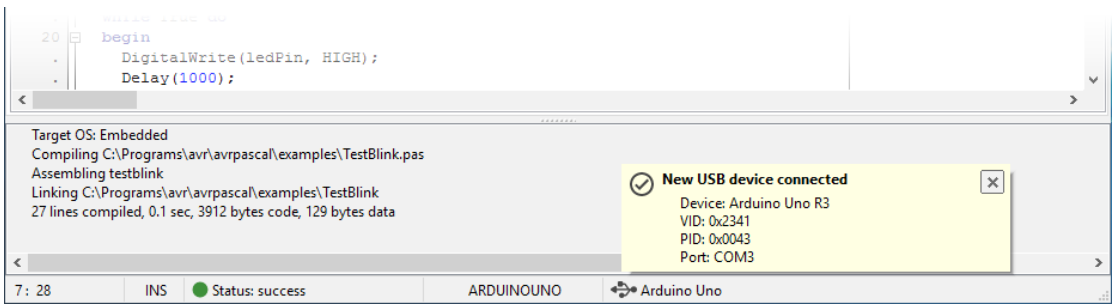


AVRPascal also allows you to import flash memory or EEPROM of the microcontroller to a file (menu *Tools->Download to File*) and export data from the selected

file to flash memory or EEPROM of the microcontroller (menu *Tools->Upload from File*). These features utilize the global microcontroller type and supported file format settings you previously configured in the *Options* window. In addition, as with uploading, there are limitations regarding the supported interfaces.



Moreover, via the *Tools* menu it is possible to read and write the fuse values of microcontrollers supporting USBasp. If they are read from the microcontroller (*Tools->Get Device Fuses*), their appropriate values (byte0/low/high/extended, among other types) will appear in the messages area of the application, while their writing (*Tools->Set Device Fuse*) is made via an extended dialog box, allowing you to change individual bits[19]. When specifying fuse-bit values other than the default ones, you should be especially careful and use the descriptions in the datasheet of individual microcontroller, because accidental settings may prevent further programming of the microcontroller using USBasp.



An additional option of AVRPascal is the detection of USB devices supported by the application (USBasp, Arduino boards). This option improves work primarily with Arduino

19 The Set Device Fuse dialog allows you to write one selected fuse at a time.

boards, in the case of using the USBasp programmer it has no major significance[20]. It can be activated in the Options window in the Uploader tab by checking *Notify the connection or disconnection of a supported device*[21]. Notifications about connecting a new device or disconnecting a previously connected one will appear just above the icon (USB connector symbol) in the status bar. Next to this icon USB devices currently connected to the computer are listed.

AVRPascal has a list of predefined USB devices with their VID and PID (vendor and product identifiers), which allow the identification of a given device as a USBAsp programmer or Arduino board. This list is also available in the *Uploader* tab and can be supplemented with your own device definitions, so that AVRPascal will be aable to detect, for example, Arduino clones. The only limitation is the PID and VID, which must be unique for each definition.

An improvement resulting from the USB device detection option, apart from the informational value itself, is the suggestion of the serial port number/name associated with the required Arduino board in the *Upload to Device* and *Download from Device* dialog windows. Thanks to this, the user does not have to select the serial port number/name from the list currently available in the operating system.

6. UnoLib library

UnoLib is a Pascal library designed for the Arduino Uno platform (which utilizes the ATmega328p microcontroller with a 16 MHz clock). This library is a translation of a subset of the standard Arduino libraries, adapted as needed for a Pascal environment. Moreover, support for fixed point numbers has been added. Modules included:

- *analog.pas* - support for analog pins
- *defs.pas* - definitions of constants, bit manipulations, port support
- *dht.pas* - support for DHT11/22 sensors
- *digital.pas* - support for digital pins
- *ds1302rtc.pas* - support for ds1302 real time clock
- *fix16.pas* - support for fixed point numbers
- *hardwareserial.pas* - support for serial communication
- *liquidcrystal.pas* - support for LCD
- *timer.pas* - time-related routines

The library modules are located in the lib directory of AVRPascal. In turn, the examples directory contains simple sample programs using UnoLib modules[22]:

- *TestBlink.pas* – turns on and off the built-in LED

20 This is due to the fact that the AVRdude program itself detects the USBAsp programmer, while in the case of Arduino boards, you must indicate the virtual serial port to which the given board is associated.

21 Activating or deactivating notifications will take effect after restarting the application.

22 Before compiling a given program, make sure that the library modules listed in the *uses* section have already been compiled.

- *TestBlinkWithoutDelay.pas* - turns on and off the built-in LED using Millis
- *TestDHT11.pas* – displays information about the temperature and humidity of the air from the DHT11 sensor on an external LCD display
- *TestDigital.pas* – turns on and off the built-in LED based on the button state
- *TestLCAutoscroll.pas* – scrolls text on the LCD display
- *TestLCBlink.pas* – displays the text “hello, world!” on the LCD display
- *TestLCChars.pas* – displays non-standard characters on the LCD display
- *TestLCCursor.pas* – turns the cursor on and off on the LCD display
- *TestLCDisplay.pas* – displays and turns off the text “hello, world!”
- *TestLCSerialDisplay.pas* – displays characters taken from the serial port on the LCD display
- *TestLCTextDirection.pas* – changes the direction of text display on the LCD display
- *TestLM35.pas* – displays the temperature value from the LM35 sensor on the LCD display
- *TestSerial.pas* – sends and receives data via the serial port

7. Updates

AVRPascal updates are regularly released on the "Electronics" page of akarwowski.pl. You can also find announcements about new versions in the "News" section of the website. Each installer includes a changelog detailing the latest improvements and fixes[23].

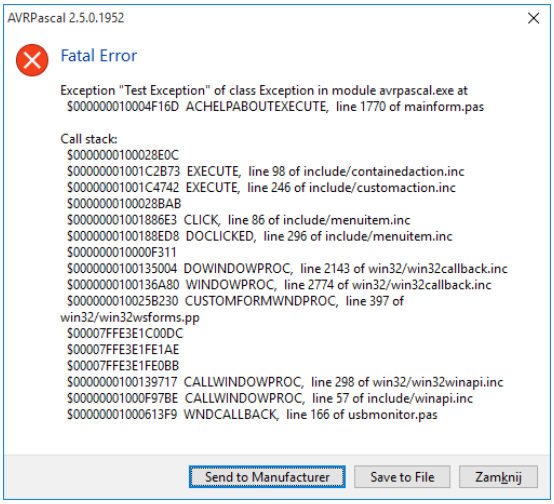
Alternatively, you can check for updates within AVRPascal itself. Go to the Help menu and select "Check for Updates." This function requires an internet connection. If a newer version is available, AVRPascal will inform you. Please note that updates are not automatic and require you to download and run the installer from the website.

8. User feedback

Users of the program can actively influence the direction of development of AVRPascal. Any suggestions and comments regarding the program's operation, including critical ones, are welcome. They can be reported by e-mail to the author at ackarwow@gmail.com.

AVRPascal can capture information about program crashes that might cause instability. These crash reports typically contain details like the location of the error and the procedures leading to it.

23 The description of current changes and improvements in the program is available on the akarwowski.pl website in the file AVRPascal_Changes.txt.



When a crash occurs, you'll see a window with the report. You can either ignore it, save it to a file for later reference, or submit it directly to the author (*Send to Manufacturer*). Sending the report requires an internet connection and will send the crash data to akarwowski.pl.

Moreover, in the *Options* window, in the *Compiler* and *Uploader* tabs, there are options to show the command line. After selecting them, the command line of the compiler and uploader will appear in the *Messages* area, respectively, with all the parameters used by AVRPascal, both user-defined and internally used. This feature can be helpful if there is a need to verify the correctness of the program's calling the compiler or uploader.

9. *Freeware* version

AVRPascal is available for free download on akarwowski.pl. It means that it can be used completely free for home use, while it is forbidden to introduce any modifications or derive financial benefits from the distribution of the program by third parties. Although, I have made my best to ensure that the program works correctly, I cannot give any guarantees for the program operation and I am not liable for any possible damage resulting from its usage.

Andrzej Karwowski

e-mail: ackarwow@gmail.com